Design and Implementation of an Active Warden Addressing Protocol Switching Covert Channels

Steffen Wendzel / Jörg Keller
(FernUniversität in Hagen)

ICIMP 2012, Stuttgart
May, 28. 2012

Outline

- Protocol Switching Covert Channels
- Bandwidth Limitation Concept
- Bandwidth Calculation
- Active Warden Implementation
- Results
- Practical Aspects
- Future Work

Covert Channel

- A communication channel that was not designed to be used for a communication
  - Presented by Lampson in 1973
- Covert channels can break mandatory security policies
  - Multi-Level Security (MLS) → Bell-La Padula
- Timing and Storage Channels
- Can be used to exfiltrate confidential data from networks

Protocol Switching Covert Channels
Two Types

Protocol Channels (information transfer based on the used network protocol)

Protocol Hopping Covert Channels (utilize multiple protocols to transfer covert channel data)

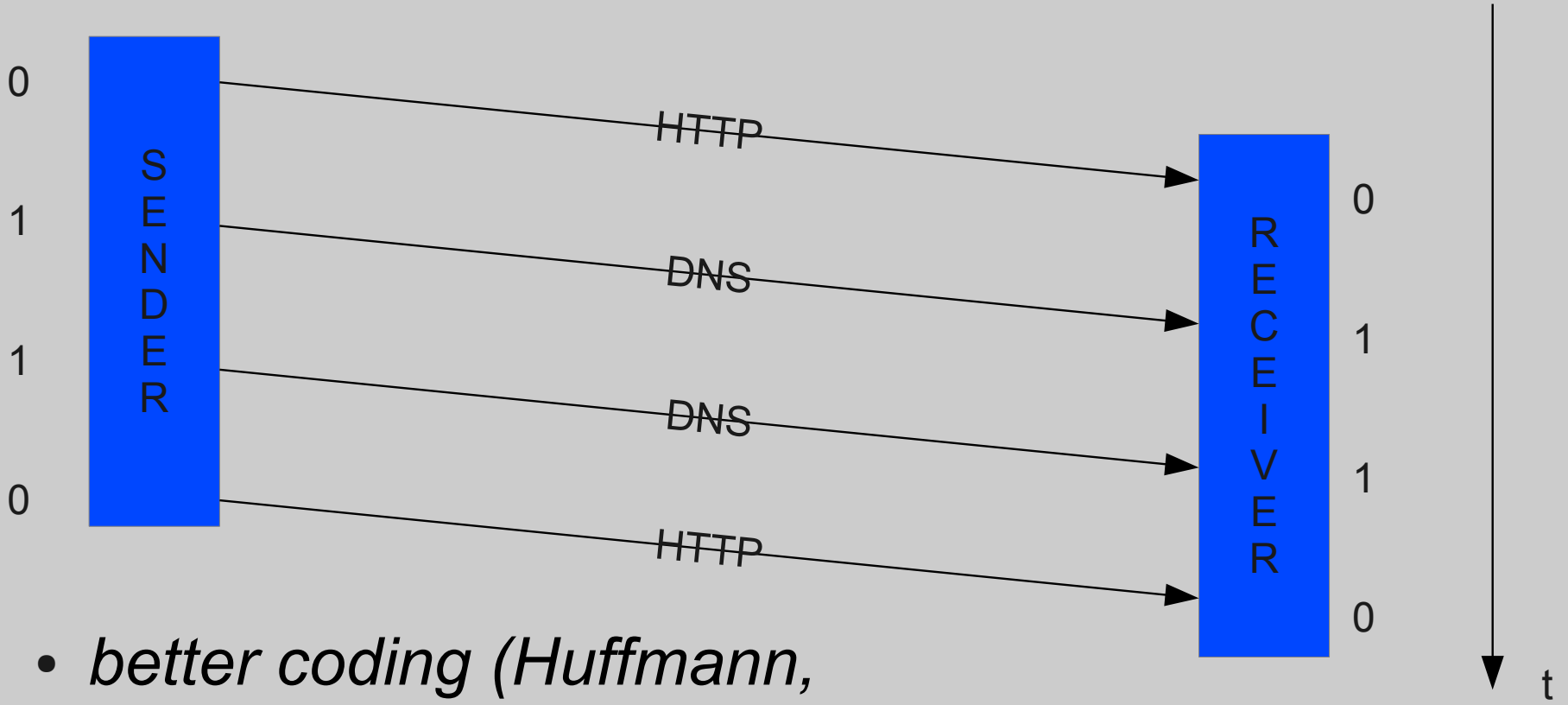
- a.k.a. „protocol switching covert storage channels“

Protocol Channels

- Can be based on different layers
  - Ethernet: *Ether Type*
  - PPP: *Protocol*
  - IPv4: *Protocol*
  - IPv6: *Next Header*
  - TCP/UDP: *Source Port* and *Destination Port*
- A protocol is linked to a bit (combination)

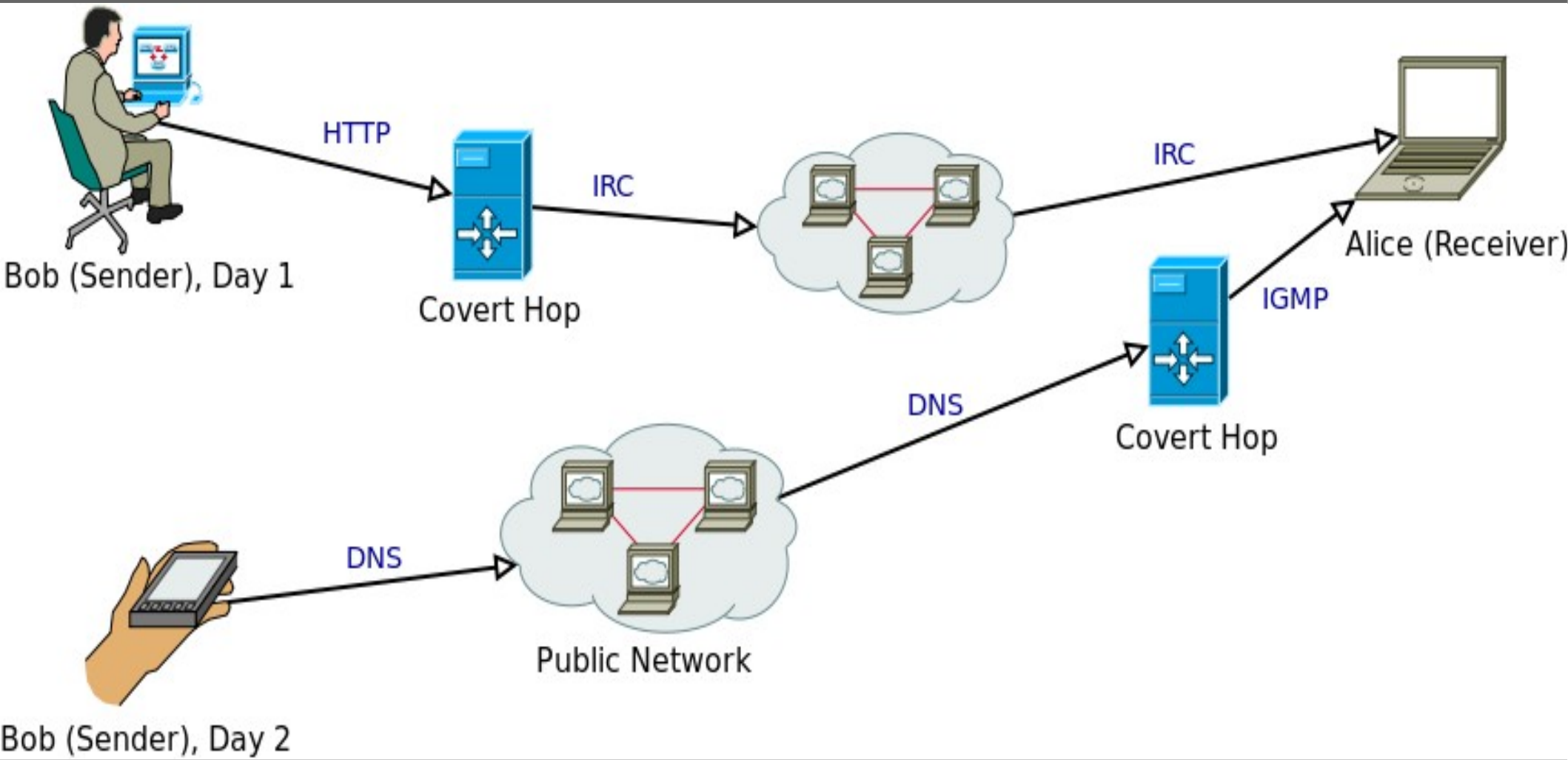
Protocol Channels

- *e.g., HTTP=0, DNS=1*

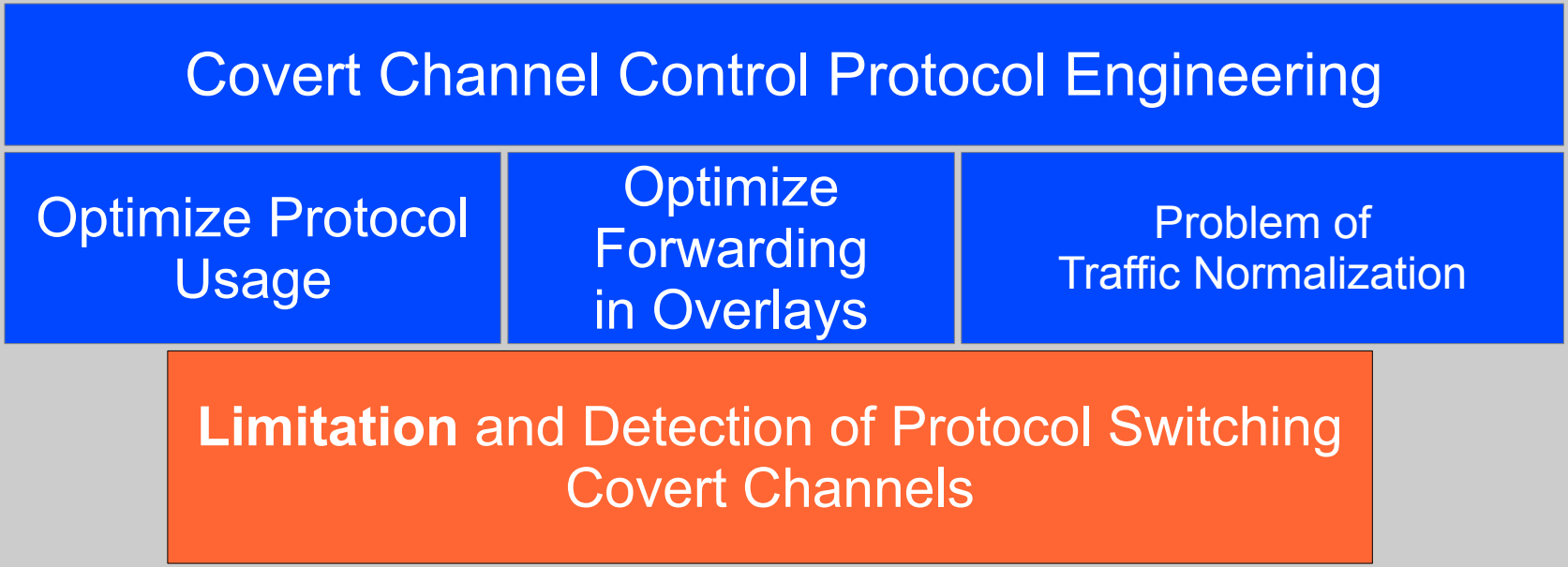


- *better coding (Huffmann, parity bits etc. possible)*

Protocol Hopping Covert Channels



Context

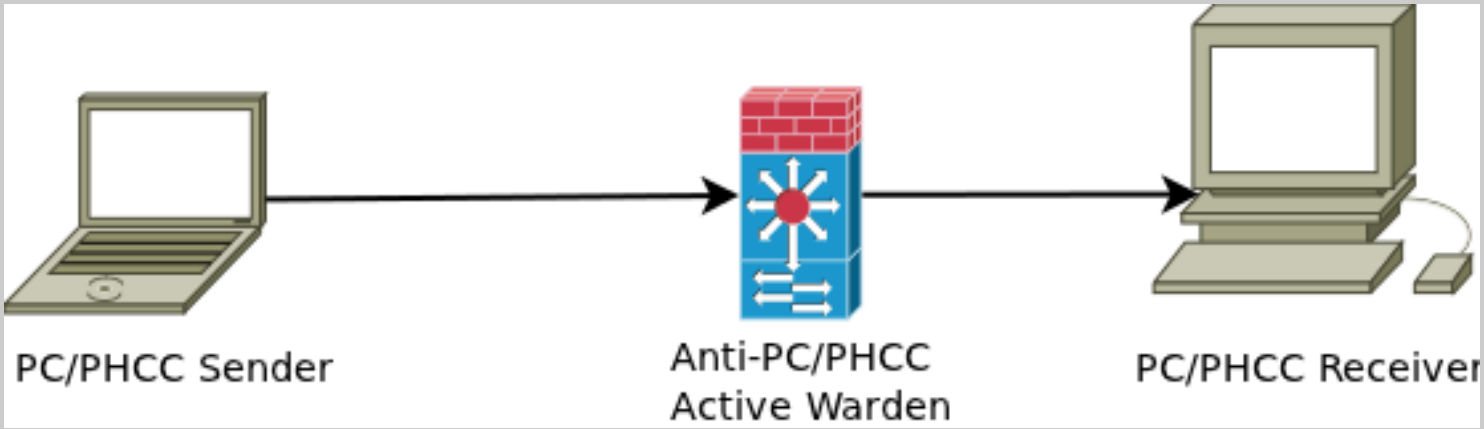

Covert Channel Control Protocol Engineering
Optimize Protocol Usage
Optimize Forwarding in Overlays
Problem of Traffic Normalization
Limitation and Detection of Protocol Switching Covert Channels

Active Warden: Concept

- First approach to counter PSCC
- Both, Protocol Channels and Protocol Hopping Covert Channels, share the capability to switch protocols.
- Slow-down protocol switches
  - … minimizing effects on policy-conform communication in the network

Active Warden: Location

- Located on the Exit Node of a LAN, e.g. Uplink of an Organization



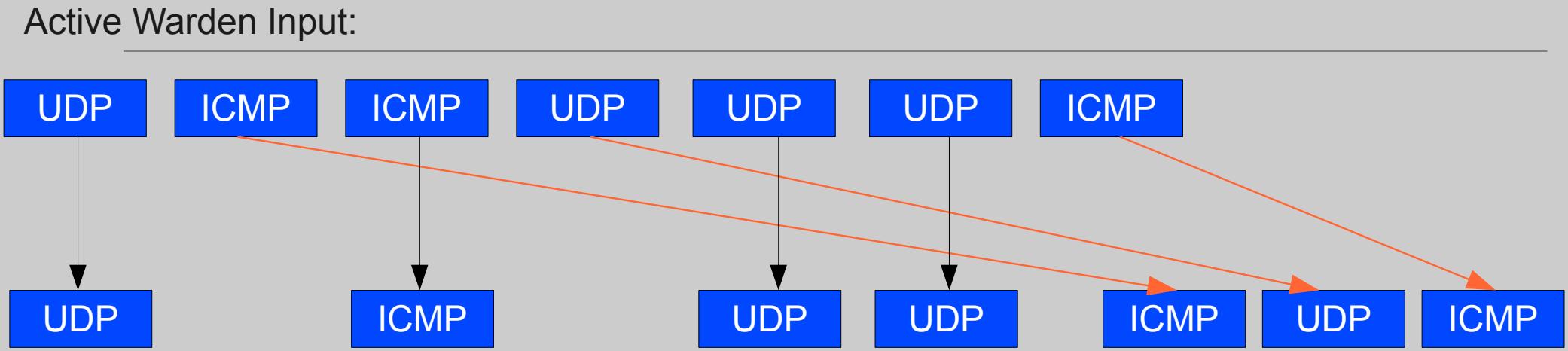
Optimization Problem

- Delay *d*
- Data leakage can occur at maximum rate *R(d)*
  - Increasing with decreasing *d*
- Side-effects for legitimate users → *S(d)*
  - Increasing with increasing *d*

- „Best“ value for *d* depends on the use-case
  - Task for the network administration

Example

- Protocol Channel based on ICMP (1) & UDP (0)
- Message „0110001“ with high *d* (e.g. *1s*)



Output: U,I,U,U,I,U,I or 0100101

Bandwidth Calculation

- Tsai/Gligor: $B = b \cdot \left(T_R + T_S + 2\mathrm{T}_{CS} \right)^{-1}$
- A Protocol Channel can transfer $b = \log_2 n$ bits per packet (*n*=#protocols used)
- Protocol Hopping Covert Channel can transfer more information per packet

Bandwidth Calculation

- Maximum possible bandwidth for PC/PHCC to prevent packet scrambling through the active warden
- Introducing *p* (probability of a protocol switch)

$$B = b \cdot \left(p \cdot d + T \right)^{-1}$$

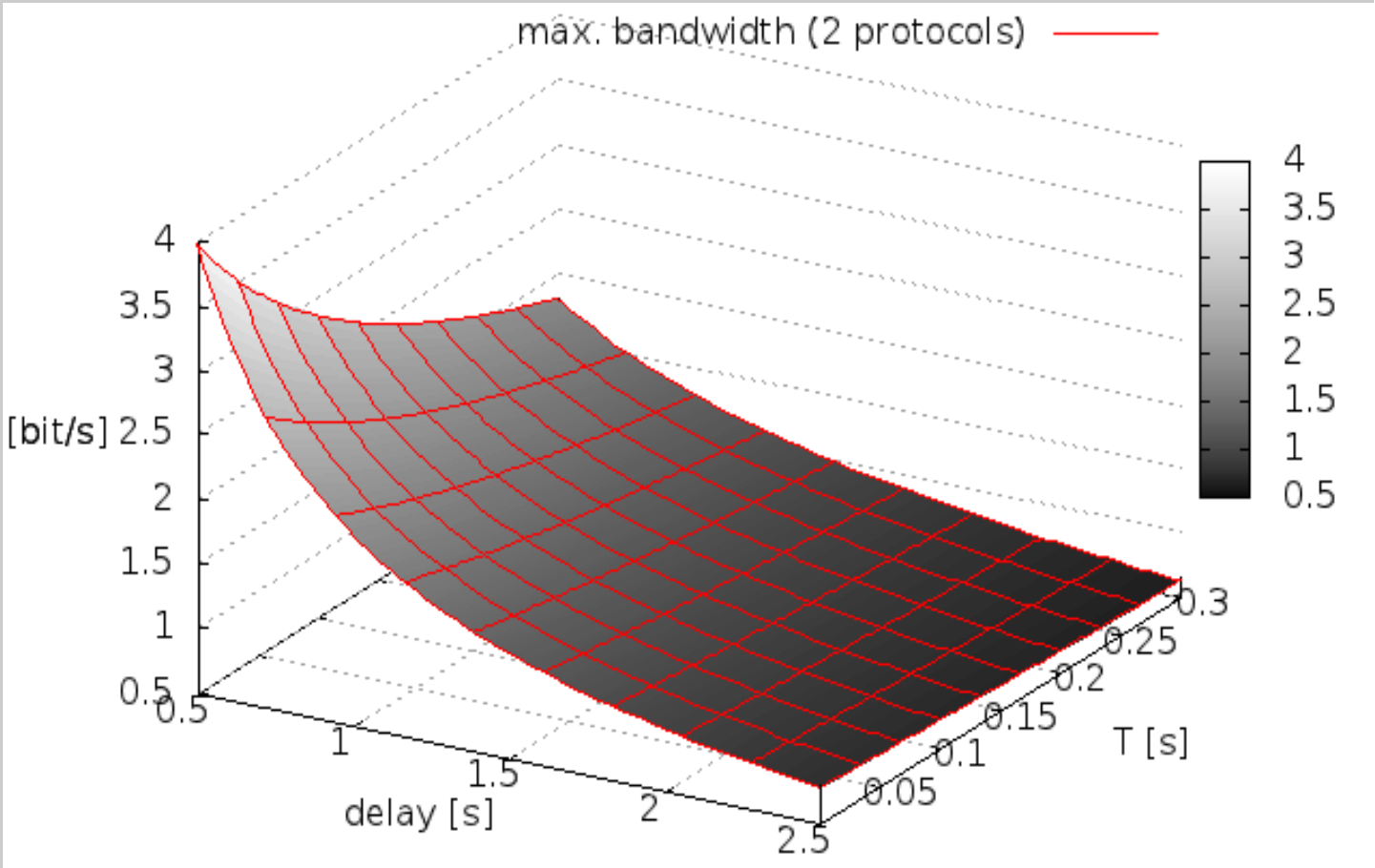
- p is (1-1/n) for n protocols (uniform coding)

$$B = b \cdot \left(\left(1 - 1/n \right) \cdot d + T \right)^{-1}$$

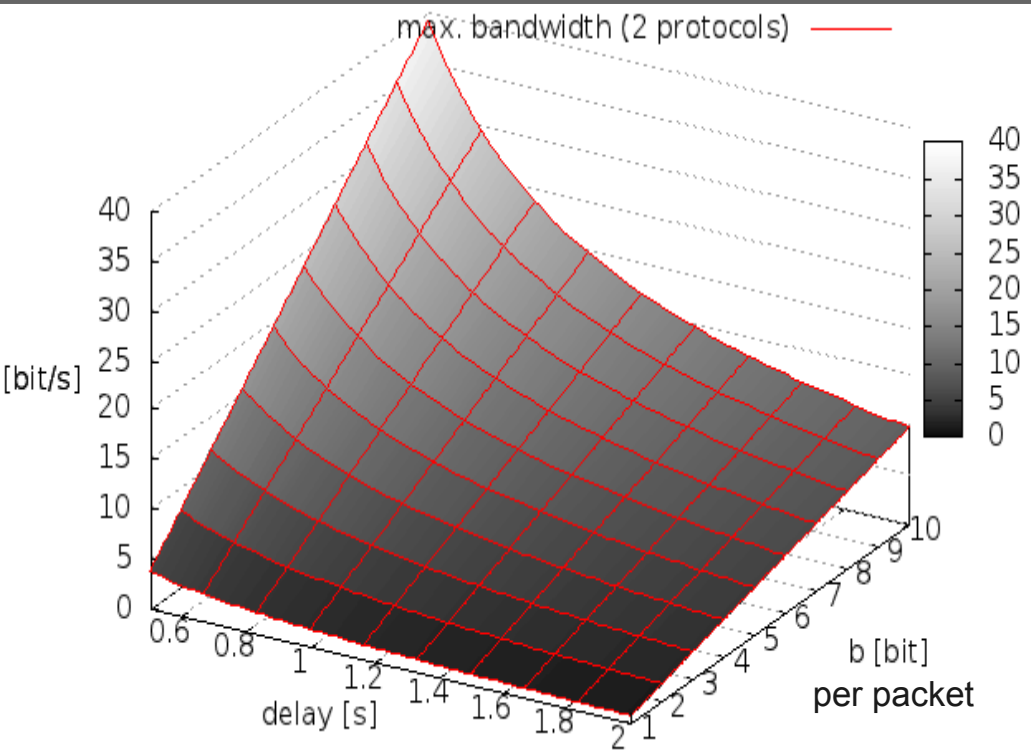
- PC with uniform coding: $B = \log_2 n \cdot \left(\left(1 - 1/n \right) \cdot d + T \right)^{-1}$

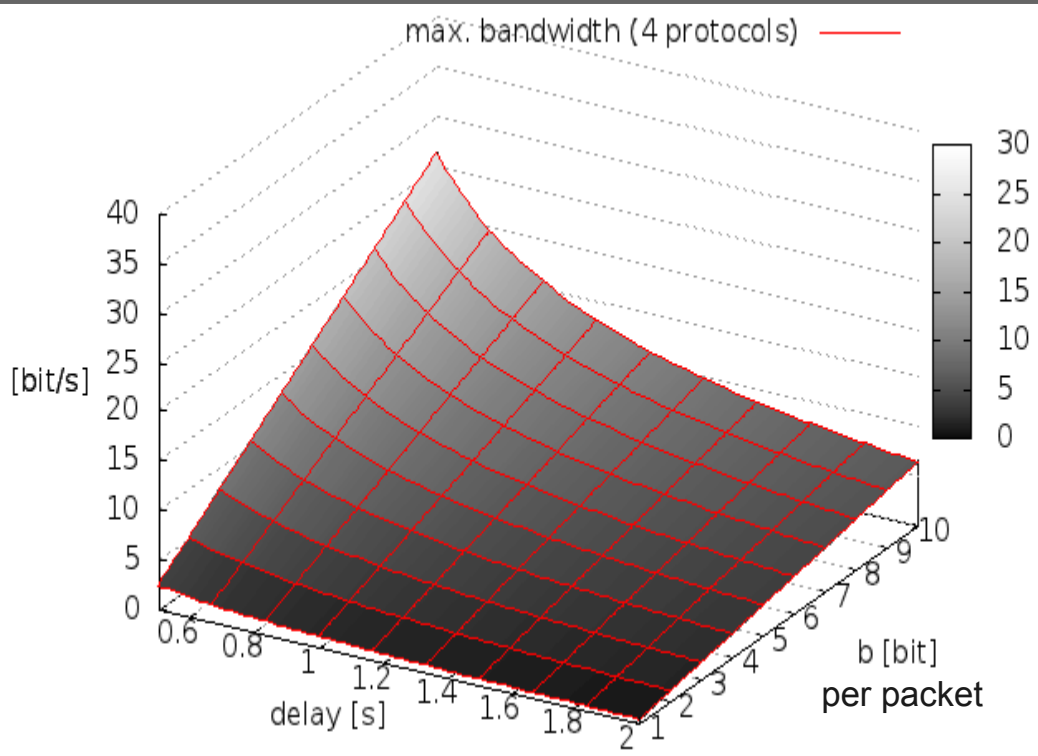
In theory ...

- … we can decrease the performance of a PC to less than 1 bit/s by introducing a delay of 2s (with realistic T)



… and for PHCC?





T=0.005, delay between 0.5 and 2s, capability to transfer 1-10 bits per packet.

→ Bandwidth decreases if the number of procotols increases

Implementation

- Two Linux 3.0 systems connected with VirtualBox
- Focus on layer 4 protocols over IPv4 only
- At the moment: only implemented for PCs
  - PHCCs can use µPs → later
- Based on a PC-Sender, the Active Warden software, and script that evaluates the correct transmission of messages.

PC Sender

- Based on „protocol channel tool“ (pct)
  - Written in Perl
  - modified: (ARP,ICMP) → (UDP,ICMP)
  - Implemented adjustable bitrate

Active Warden

- Based on delay-net
  - Perl based as well and written by Berrange
  - Uses QUEUE feature of Netfilter/iptables
    - QUEUE redirects packets to a userspace program
    - Delay-net introduces configurable delays using the IPQueue module
    - Original goal is to to simulate WAN environments
  - Modified delay-net to introduce delays only if protocol switches are taking place

Results

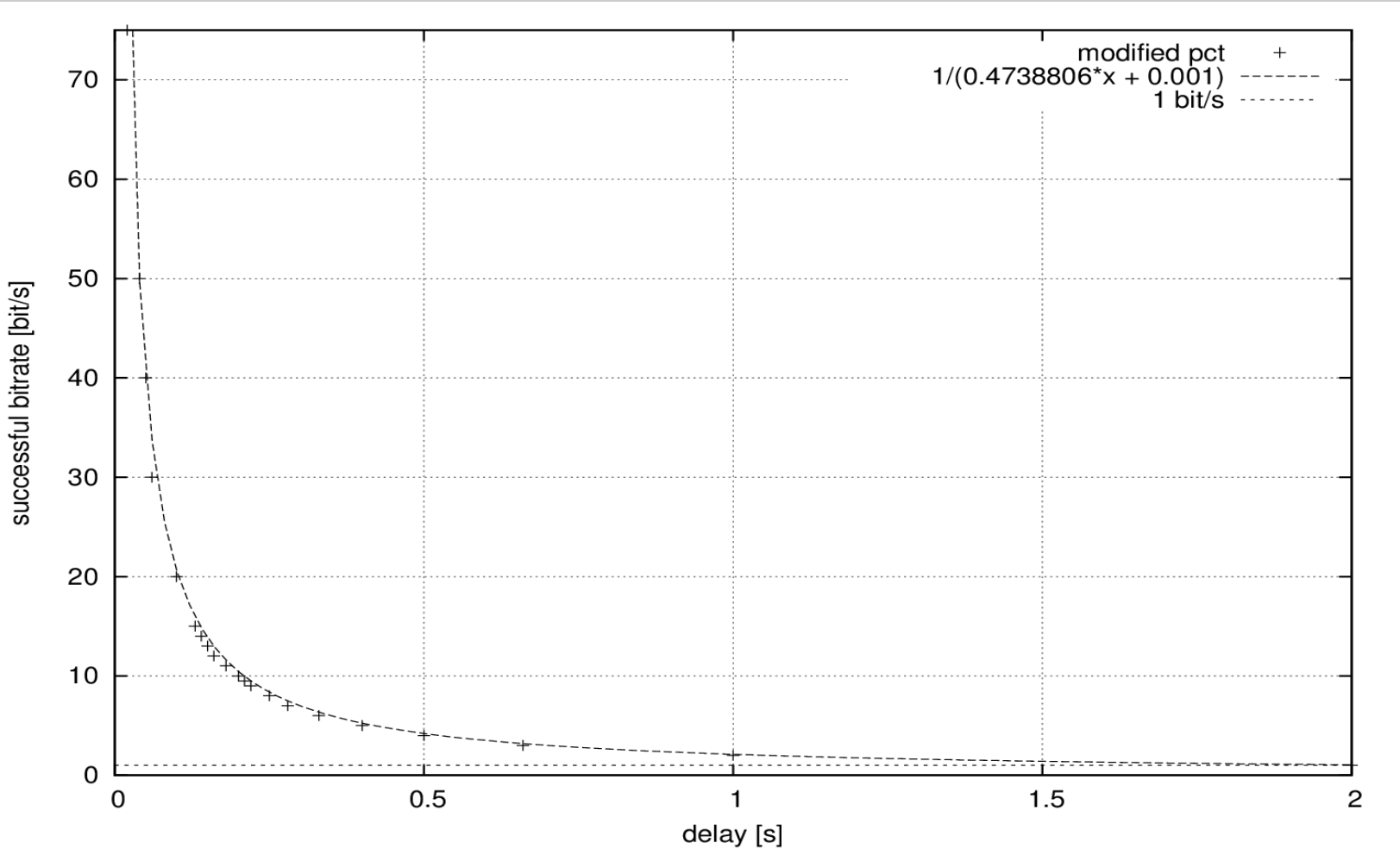
- We measured T=0.005s (in average)
- If two protocols are used, p was approx. 0.473 but in a real (non-simulated) network it was 0.53

Theory meets Reality

- T=0.005, d=2.1s → bandwidth limited to 1 bit/s
- T=0.005, d=1s → bandwidth limited to 2.088 bit/s

Estimated bandwidth of our formula in comparison to pct's actual bandwidth within our virtual network

→ no substantial differences



Improved Coding for PCs

- However, the bandwidth can be increased if better codings are used
- e.g. error correcting codes
- Uniform codings are realistic (encrypted payload)

Improved Coding for PCs

- Good choice: Only send packets if a different bit value should occur (000001111110000 results in only 3 packets).
  - Is a combination of a PC and a timing channel (measures timing differences between packets)
- Only the whole message is delayed by *d*
- **Solution:** Randomized delay

Micro Protocols for PHCC

- PC results are nice, but what about PHCC in practice?
- PHCCs can contain small covert channel-internal control protocols (*micro protocols*)
- Micro protocols can contain sequence numbers
- The PHCC receiver can sort received packets
  - Our active warden would be useless in that case
- However, we force the PHCC to use sequence numbers → reducing payload bits/packet!

Practical Aspects

- DNS ↔ HTTP(S) / Multi-protocol server (e.g. SMTP+POP3)
  - Special rule
- Multiple senders → NAT → many protocol switches for „one“ host (the NAT gateway)
  - White-listing (bad) or Remote physical device fingerprinting (*pointed out by reviewer*)

Practical Aspects

- Redundancy
  - Like every firewall: A single-point of failure
  - Solution: OpenBSD CARP-like functionality

Practical Aspects

- End-User Limitation:
  - No extensive end-user study
  - Measured HTTP request/response times
  - 10 Mbytes download
  - 0.41-0.57s without active warden
  - Active warden, 0.25 bit/s protocol channel running in parallel: 0.43-3s
  - 4s rule for website rendering by Akamai (*pointed out by reviewer*)

Conclusion

- Presented the first Active Warden able to limit PSCCs (protocol channels as well as protocol hopping covert channels) by introducing delays …
- … and seems to be practically usable for high-security environments.
- However, the NAT aspect and PHCCs with micro protocols seem to be challenging problems to solve.

Future Work

- Detection of Protocol Channels and Protocol Hopping Covert Channels
  - Delay detected PC/PHCC communications instead of all communications

    → additionally minimizes effects for end-users

Are there any Questions?

Image Source: Amazon